Création automatique d'atlas de surfaces 3D en utilisant un algorithme de segmentation par seuillage récursif.

Rodolphe Charrier

Création automatique d'atlas de surfaces 3D en utilisant un algorithme de segmentation par seuillage recursif.

Rodolphe CHARRIER

mars-août 1993

UNIVERSITE PARIS XI – (ORSAY)
D.E.A. Informatique 92-93

INRIA

MÉMOIRE D.E.A.

Sujet : Création automatique d'atlas

Directeur de Recherche : A. Gagalowicz

Candidat : Rodolphe Charrier

Remerciements

Je tiens à remercier Mr André Gagalowicz pour m'avoir acceuilli dans son équipe et pour avoir suivi efficacement ce stage.

Je remercie également mon camarade stagiaire Vincent Balard qui a eu la patience de m'expliquer un certain nombre de détails concernant les atlas.

Je remercie Hussein Yahia pour son aide precieuse.

Table des matières

Table des figures

Chapitre 1

Introduction

L'utilisation des textures est très répandue en synthèse d'image pour accroître le réalisme. Le sujet de cette étude ne traite pas directement des textures mais de la façon dont on peut les plaquer le mieux possible sur la surface d'un objet. La solution est de découper cette surface en autant de parties qu'il sera nécessaire pour avoir une texture non déformée : il s'agit de la création d'atlas.

Dans un premier temps, nous expliquerons les notions de texture, d'atlas, et le but poursuivi. Puis, deux types d'algorithme seront analysés : l'un de type ascendant, l'autre descendant et là aussi nous expliciterons davantages ces termes. Enfin, l'exploitation des résultats nous amènera à considérer certaines améliorations et quelques perspectives de recherche.

Chapitre 2

Cadre du probleme et définition des atlas

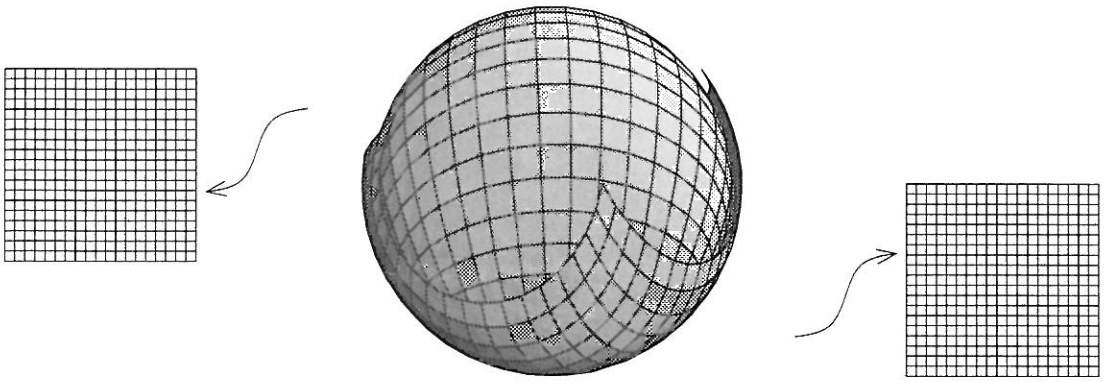
Dans ce chapitre, nous allons définir la notion d'atlas et montrer son intéret en image de synthèse.

2.1 utilisation intuitive des atlas pour la projection de textures

Un objet réel n'est pas seulement reconnaissable visuellement par sa forme, ses caractéristiques géométriques, mais aussi par une caractéristique liée à sa texture qui peut être selon les cas une matière ou un habillage de l'objet. Un morceau de bois par exemple a une texture de type volumique représentée par les nervures qui le parcourent. La modélisation de cette texture se fait en bloc, de façon volumique, la texture étant définie en chaque point du volume. Mais nous n'entrerons pas dans les détails de ce type de textures.

Les textures qui nous intéressent ici sont les textures surfaciques planes qui servent à habiller l'objet, tout comme un couturier habille sur mesure un client. Il s'agit plus généralement de plaquer un motif sur un objet comme pour le tapisser. Cette technique est plus délicate car elle doit tenir compte de la géométrie à la surface de l'objet.

En effet, le plaquage de textures surfaciques planes fonctionne de la façon suivante. On fait la mise en correspondance des points d'une surface 3D avec ceux d'une image de la texture. Cette mise en correspondance est appelée fonction de mapping. Chaque point de la surface de l'objet étant ainsi relié à un point de l'image de la texture on peut ensuite "plaquer" cette image sur la surface. Arrêtons nous cependant quelques instants sur cette manipulation : l'image de la texture est elle-même définie sur une surface simple.



Un exemple d'atlas à deux cartes défini sur la sphère unité. Le domaine de définition de la carte du dessus a été représenté avec un rayon légèrement inférieur pour mieux distinguer chacune des parties.

FIG. 2.1 - Un exemple de cartes.

Ou bien on la définit sur un plan, auquel cas la fonction de mapping sera une projection du plan sur la surface, ou bien sur un cylindre et on a alors une projection cylindrique, ou bien encore sur une sphère et on effectue une projection sphérique.

Mais cette technique comporte un inconvénient majeur : si pour certaines textures uniformes et peu régulières, les déformations ne sont pas visibles, pour des textures périodiques comme celle du damier, les déformations dues à la projection sont trop nettes pour en faire un objet réaliste. Il faut donc améliorer cette projection ou faire qu'elle soit plus sélective vis à vis de la surface.

C'est justement ce qui est proposé avec la création d'atlas. On cherche à découper la surface de l'objet en différentes cartes homéomorphes au plan. On plaque ensuite la texture sur chaque carte (projection) pour minimiser les déformations. Cette façon d'aborder le problème n'est pas sans rappeler les procédés de fabrication du prêt-à-porter : une chemise n'est autre que l'assemblage par couture d'un ensemble de morceaux de tissus prédécoupés (les patrons du vêtement). Représentons-nous ces morceaux comme des cartes dont l'ensemble forme l'atlas du vêtement.

2.2 définition des atlas

Mais revenons à des définitions plus mathématiques de ces notions très intuitives.

La notion de surface dans un espace de dimension trois est un cas particulier de ce que les géomètres appellent une sous-variété plongée. Nous allons préciser quelques définitions, et montrer comment on peut utiliser ce formalisme pour résoudre notre problème de mapping.

2.2.1 Définitions

Une façon de définir une surface est de recourir à un système de coordonnées locales. Une surface est, par définition, un ensemble de points de $\mathbb{R}^3$ homéomorphe au voisinage de chaque point à un ouvert du plan. Chacune des applications continues Φ_i d'un voisinage U_i d'un point M_i de la surface vers un ouvert $\Phi_i(U_i)$ du plan s'appelle une *carte*. On appelle *atlas* de classe C^n un ensemble de cartes vérifiant les propriétés suivantes :

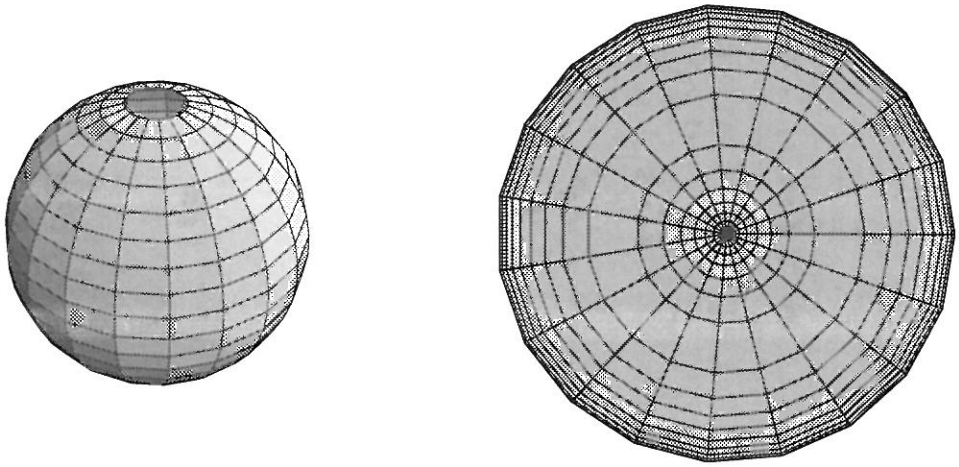
1. L'atlas $\mathcal{A}$ possède un nombre dénombrable de cartes Φ_i.
2. Chacune des cartes Φ_i est un difféomorphisme de classe C^n.
3. Pour tout couple de cartes Φ_i, Φ_j les applications $\Phi_i \circ \Phi_j^{-1}$ et $\Phi_j \circ \Phi_i^{-1}$ sont des difféomorphismes de classe C^n entre $\Phi_i(U_i \bigcap U_j)$ et $\Phi_j(U_i \bigcap U_j)$.

Une surface abstraite est alors définie par la réunion $\Sigma = \bigcup_i U_i$. Les applications $\Phi_i \circ \Phi_j^{-1}$ s'appellent des changements de carte. Une surface concrète est un *plongement*[1]. Dans le cas qui nous intéresse, cet espace sera $\mathbb{R}^3$. On peut alors parler de structure différentielle sur la surface. Ces rappels n'ont pour but que de replacer les définitions dans leur contexte, car pour suivre cet exposé, il n'est pas necessaire d'avoir de notions précises de géométrie différentielle.

La notion de cartes et d'atlas a été introduite en géométrie différentielle pour pouvoir manipuler des objets dont les propriétés sont proches du plan (C^n difféomorphisme) au voisinage de chaque point, mais pour lesquels on ne peut pas définir de correspondance globale. Une sphère est un exemple simple de surface que l'on ne peut définir à l'aide de moins de deux cartes (voir la figure 2.1). Ce sont donc des contraintes topologiques qui ont abouti à ce formalisme.

En ce qui concerne le problème du mapping de texture, les contraintes sont plus fortes. De façon évidente, il n'est pas possible de définir une fonction de mapping globale pour une sphère à cause de problèmes topologiques, mais même pour une sphère privée d'un point (et donc homéomorphe au plan), la meilleure fonction de mapping déforme

[1]Ou plus généralement une *immersion*. Une immersion est une application dont la différentielle en tout point est injective, un plongement est en plus injectif. L'immersion n'assure en fait la propriété d'injection que localement. On pourra comparer les deux définitions dans [Spi79, pages 61 et 65] d'une surface abstraite dans un espace vectoriel [Hir76, chapitre 1][Spi79, chapitre2, tome1]



La portion de sphe're. Image de la surface par la carte.

FIG. 2.2 - Un exemple de carte sur la sphe're unite' prive'e d'une petite calotte.

nécessairement la texture de façon importante au voisinage du point manquant. On peut voir sur la figure 2.2 que le bord du domaine de définition de la carte est très compressé orthogonalement aux rayons par rapport aux polygones correspondants sur la sphère. Et il n'est pas possible d'améliorer sensiblement la déformation sans augmenter le nombre de cartes, ce qui introduit de grandes discontinuités texturelles.

Cependant nous n'avons pas besoin d'une notion de continuité aussi forte que celle qui est imposée par les géomètres. En particulier les changements de carte ne présentent pas d'intérêt pour le plaquage de texture. Nous allons donc modifier les définitions pour les adapter à notre problème. Nous avons choisi de conserver le même vocabulaire par souci de clarté, mais il faudra conserver à l'esprit que ces mots prennent ici un sens légèrement différent.

2.2.2 Adaptation de la notion d'atlas au problème du mapping.

Les couturiers utilisent couramment la segmentation de surfaces lorsqu'ils mettent à plat un vêtement sous forme d'un patron, dont la définition se rapproche de très près de celle des atlas. Ils peuvent ainsi obtenir des surfaces quasiment développables localement, mais bien sûr pas globalement. Si l'on remarque que l'on peut approcher toute surface suffisamment régulière par une surface polyédrique, on s'aperçoit a fortiori que l'on peut approximer, pour une distance maximale donnée, les surfaces que nous

avons à manipuler par des surfaces développables[2] par morceaux.

Nous allons modifier légèrement la notion d'atlas pour mieux répondre au problème qui nous intéresse. Il faut d'abord remarquer que la fonction de mapping doit associer à chaque point de la surface un point unique dans le plan de la texture. Pour lever l'indétermination de la carte à choisir en un point donné de la surface, on définit un ensemble $\widetilde{U_i}$ d'ouverts, inclus dans les U_i, dont les intersections deux à deux sont vides et dont la réunion des adhérences recouvre toute la surface. Les cartes restreintes aux $\widetilde{U_i}$ forment bien une fonction de mapping, sauf pour les bords de cartes. Pour les points sur ces derniers, il faut préciser à quelle carte on considère qu'il appartiennent lors du rendu.

En fait, l'algorithme de segmentation étudié par la suite détermine, non pas les cartes qui sont mathématiquement des applications , mais les ouverts du plan associés, à savoir les régions obtenues après segmentation, que nous nommerons desormais par abus de langage des cartes.

Abordons maintenant l'explication de l'algorithme qui va nous permettre d'extraire ces cartes du modèle géométrique de l'objet quelconque étudié.

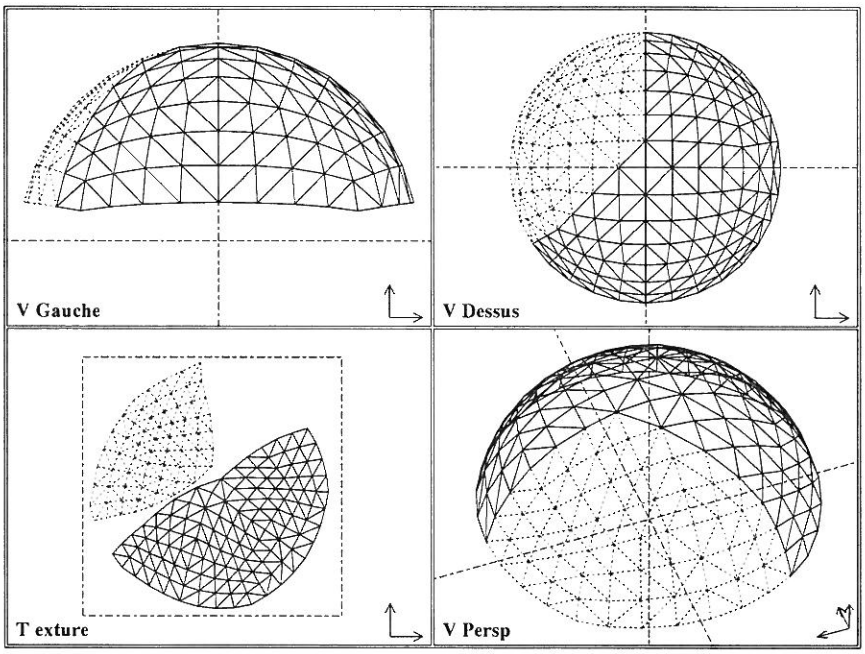
---

[2]une surface est dite *de'veloppable* si sa courbure Gaussienne est identiquement nulle. Ceci correspond au fait qu'il existe une isome'trie qui associe une portion de plan a' cette surface. Au travers de cette isome'trie, il est possible de transporter une image bidimensionnelle sur la surface, sans la de'former. Ainsi, du point de vue qui nous inte'resse, la classe des surfaces de'veloppables est identique a' la classe des surfaces posse'dant un "mapping" sans distorsion. Une telle surface est un cas particulier des surfaces re'gle'es dont l'e'quation ge'ne'rale est de la forme :

$$\Sigma(s,t) \quad = \quad \alpha(t) + sw(t) \tag{2.1}$$

La nullite' de la courbure de Gauss s'exprime alors par la formule [DoC76] :

$$\det\left(\frac{\partial \alpha}{\partial t}, w(t), \frac{\partial w}{\partial t}\right) \quad = \quad 0 \tag{2.2}$$



On a représenté ici une calotte sphérique composée de deux cartes. Le carré dans la vue "Texture" représente la taille de l'image dans laquelle seront découpés les motifs.

FIG. 2.3 - Organisation de l'écran du modeleur ACTION 3D.

Chapitre 3

Présentation des deux algorithmes de segmentation utilisés

Ce chapitre est consacre à l'étude de deux algorithmes de segmentation qui seront comparés : l'un d'eux a déjà été implémenté par Jérome Maillot dans sa thèse sur les textures faite au sein de l'équipe SYNTIM et soutenue en mai 1992 ; l'autre est l'aboutissement de ce stage.

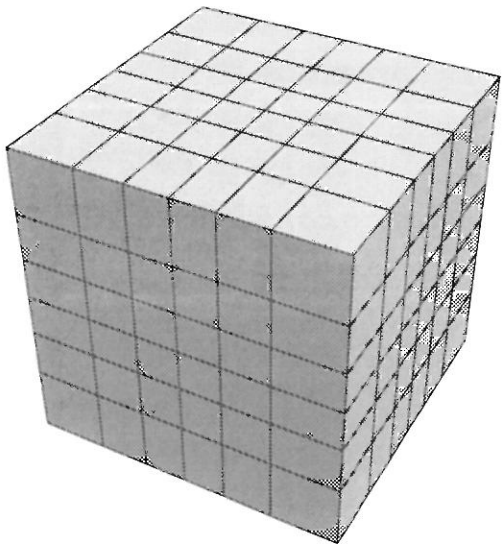
3.1 Approche ascendante

Nous ne n'expliquerons pas en détail cette méthode qui a déjà été traitée, voir [Mai92] pour plus d'informations. On n'en trouvera ici que les grandes lignes, qui donneront un aperçu suffisant pour la comparaison des résultats. Dans toute la suite, nous supposerons que notre surface est définie par une approximation polygonale.

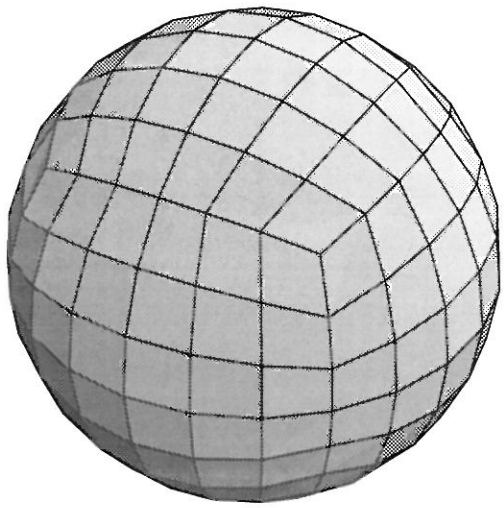
3.1.1 Bucketisation de la surface

Si l'on se place sur la sphère de Gauss $\mathcal{S}^2$, on peut ramener le problème de partitionnement à un problème de calcul de densité. La sphère unité est associée à la surface par l'application $\underline{N}$ qui fait correspondre à chaque point de la surface sa normale. On pourra se référer à [DoC76, chapitre 3] pour plus de précisions. Notre surface étant polygonalisée, on définit l'application de Gauss par un ensemble de vecteurs pondérés par l'aire de la face dont ils proviennent. La méthode est basée sur une discrétisation de la sphère $\mathcal{S}^2$ en *buckets*. L'intérêt est que l'on n'aura pas besoin de plus de quelques centaines de *buckets* pour classer les normales, quelle que soit la complexité de la surface.

Il faut prendre un recouvrement homogène et symétrique de la sphère. La



Quadrillage du cube unité.



Sa projection sur la sphère $\mathcal{S}^2$.

La sphère discrétisée avec un paramètre de subdivision 6 est découpée en 216 régions dont la surface varie entre 0.05 et 0.07, et dont l'angle maximal d'ouverture est compris entre 19°et 23°.

FIG. 3.1 - Discrétisation de la sphère unité.

discrétisation n'est donc pas issue d'une découpe par les parallèles et les méridiens mais d'un partage basé sur la projection d'un cube sur la sphère tel que nous l'avons représenté sur la figure 3.1. On quadrille chaque face du cube unité et on projette la grille ainsi obtenue sur la sphère. Pour éviter une trop grande disparité dans la taille des régions, le quadrillage sur le cube n'est pas pris régulièrement, mais avec une loi en $\tan(x\frac{\pi}{4})$, $x \in [-1, 1]$ ce qui assure un échantillonnage uniforme le long des 12 arêtes du cube une fois projetées sur la sphère. La figure 3.1 donne un exemple de discrétisation. Ce genre de découpe permet de calculer directement et rapidement le numéro du *bucket*[1] contenant une direction[2]. On remarquera aussi que l'on sait déterminer simplement le nombre et les numéros des *buckets* voisins.

Cette technique permet de mettre en évidence assez simplement la direction de projection qui sera utilisée ensuite pour la mise à plat des cartes en évitant des distorsions trop grandes.Dans l'exemple de la figure 3.1, un angle maximal de distorsion de 40°donne des calottes correspondant à peu près à quatre *buckets.*

Un dernier problème est celui de la connexité des cartes. En effet, la densité de normales obtenue au travers de l'application $\underline{\mathrm{N}}$ ne contient plus aucune notion de voisinage. Le problème se pose par exemple pour une surface périodique : dans ce cas toutes les faces distantes d'une période se projettent au même endroit sur $\mathcal{S}^2$ mais proviennent de régions qui peuvent ne pas être connectées. Il est donc possible d'obtenir des zones de forte densité sur $\mathcal{S}^2$ de manière fortuite, car composées de normales issues de faces éloignées mais d'orientation proche (comme sur la surface de la figure ??).

Pour contrecarrer cet effet, on travaille avec des piles de *buckets*, dont chaque élément correspond à une région connexe de la surface, tous les éléments de la pile représentant une même direction. Le remplissage des *buckets* se fait de la manière suivante :

- On initialise le système avec un *bucket* par direction, le nombre de directions est déterminé par le degré de subdivision de la sphère.
- On stocke chaque numéro de face dans son *bucket.*
- Pour chacun des *buckets* initiaux, on effectue un parcours par adjacence, de manière à séparer les composantes connexes. Chaque nouvelle composante donne naissance à un autre *bucket* de la même pile. On conserve la liste des faces du bord en fin de chaque parcours.
- A l'aide des faces de bord, on construit un graphe d'adjacence sur les *buckets.* On élimine aussi à ce moment les *buckets* vides ou peu remplis.

[1] Ce terme est utilisé en segmentation d'image pour représenter des petites régions considérées comme uniformes, et dont tous les pixels subissent un traitement identique.

[2] Il suffit en fait de projeter les normales sur le cube, et de regarder dans quelle face et dans quelle cellule on aboutit.

On peut assimiler cette méthode à un lissage préalable de la surface. On crée un nouvel objet formel à basse résolution qui va servir de base de départ aux algorithmes.

3.1.2 Mise en place de l'algorithme.

L'algorithme effectue une segmentation de l'ensemble des buckets qui utilise un critère de similitude. On dispose également d'un graphe d'adjacence des buckets, $\mathcal{G}_\beta$.On considère que deux buckets sont adjacents de's que les re'gions correspondantes sur l'objet le sont et on prend comme caracte'risation de l'adjacence des re'gions le fait qu'elles aient au moins une arête commune.

Pour simplifier le propos, on appelle le critère d'adjacence une *distance* entre *buckets* (même si ce n'est pas une distance au sens mathématique du terme). Cette distance est une sorte de différence entre les normales moyennes de deux *buckets* adjacents, voir [Mai92].

E'tant donne' un seuil d, on obtient une segmentation de l'objet en fusionnant successivement toutes les paires de re'gions dont la distance est plus faible que d. Cette segmentation est donc bien de type ascendant puisqu'on effectue des fusions de *buckets*.

Pratiquement, dans le modeleur ACTION 3D, l'opération est une boucle interactive paramétrée par le nombre de régions. Avec le bouton gauche de la souris on détermine ce nombre (il est affiché à l'écran), dans le même temps, les régions obtenues sont représentées en fausses couleurs sur l'objet, puis le relachement du bouton effectue la mise à plat des cartes de l'atlas fabriqué.

On voit donc ici que la segmentation va du simple au composé. L'algorithme développé dans la prochaine section évolue à l'inverse de façon descendante.

3.2 Approche descendante : segmentation par seuillage récursif sur les régions

3.2.1 Intérêt de l'approche descendante

- Les calculs sur l'objet polygonalisé sont d'abord globaux pour devenir progressivement locaux.
- La localisation progressive du traitement permet de restreindre les divisions de régions et de controler la segmentation.
- On dispose d'informations exploitables dès les premières étapes: en effet, les premières segmentations contiennent des régions déjà très homogènes en valeur

de courbure: les régions planes pourront être déterminées dè la première segmentation, ce qui est moins fréquent dans les segmentations ascendantes (voir le chapitre des résultats).

- Enfin, nous pensons qu'intuitivement, la segmentation par division récursive ressemble plus à la tendance qu'a l'humain à chercher, progressivement, plus de détails dans ce qu'il regarde.

3.2.2 Division récursive de régions

Plusieurs méthodes d'extraction récursive de régions ont déja été développées, non pas en image de synthèse mais en traitement d'images. L'algorithme développé par la suite utilise le seuillage de Kohler qui n'a été implémenté jusqu'à présent que sur des images, voir [Koh81]. On a adapté l'algorithme initial prévu pour traiter des pixels, c'est-à-dire trois valeurs de niveau de gris en RVB pour un point de l'image (donc trois composantes pour un élément de base: le pixel). Comme on travaille la plupart du temps en image de synthèse sur des objets polygonalisés, l'élément de base sera ici une facette du polygone-objet et les parametres seront les deux valeurs de courbure associées, courbure minimale et courbure maximale (voir le calcul de la courbure en annexe).

3.2.3 Seuillage de Kohler

Approche mixte Contour-Région

La méthode de seuillage de Kohler, voir [Koh81], est basée sur l'heuristique ($\mathcal{H}$) 3.2.3.1 suivante:

Heuristique 3.2.3.1 *Les contours correspondant à de réelles discontinuités sur l'image ont tendance à marquer un fort contraste, tandis que ceux qui sont peu contrastés ne correspondent pas à de réelles frontières de régions.*
Ceci pour les images, par conséquent en image de synthèse:
le seuil optimal pour la segmentation d'un objet est celui qui détecte plus d'arêtes franches bien contrastées, i.e entre deux faces adjascentes la différence de courbure est grande, et moins d'arêtes peu contrastées qu'un autre seuil.

C'est une heuristique simple et évidente sur laquelle reposent la plupart des segmentations par détection de contours en traitement d'images.
Désormais, nous utiliserons le langage suivant:

- une carte est un ensemble connexe de faces sur l'objet.

- une frontière est l'ensemble des arêtes frontières à une région
- le contraste est la différence entre deux valeurs de courbure.

Calcul de la fonction de Contraste Moyen observé

On appelle $AVCT(s)$ le contraste cumulé moyen observé aux abords de toutes les frontières détectées par un seuil s. $AVCT(s)$ est calculé comme suit: considérons une frontière entre deux faces adjacentes (qui ont une arête commune) F_a et F_b, de courbures respectives (courbure minimale ou courbure maximale) C_a et C_b, avec $C_a <= C_b$ par exemple.
Alors tous les seuils s tels que $C_a \leq s < C_b$ peuvent détecter cette frontière.

On peut alors, déterminer le nombre de frontières dans l'objet qu'un seuil donné s peut détecter.

$$N(s) = \sum_{pour P_a, P_b adjacents} p(C_a, C_b, s)$$

avec

$$p(C_a, C_b, s) = \begin{cases} 1 & \text{si } (C_a \leq s < C_b) \\ 0 & \text{sinon} \end{cases}$$

On peut également déterminer le contraste absolu total observé sur toutes les frontières détectées par s, en sommant les contrastes locaux cl, avec:

$$cl(C_a, C_b, s) = |C_a - C_b| \text{ si } s \text{ appartient à } [C_a, C_b]$$

En fait, d'éventuelles opérations de correction peuvent modifier les courbures des faces, notamment sur les frontières donc sur les bornes des intervalles $[C_a, C_b]$. Toutes les valeurs de $[C_a, C_b]$ ne constituent donc pas des seuils équiperformants.
On privilégie donc le centre de $[C_a, C_b]$ en utilisant une fonction $rc(s)$, de contraste *relatif à s.*

$$rc(C_a, C_b, s) = MIN(C_b - s, s - C_a) \text{ pour } s \text{ appartenant à } [C_a, C_b]$$

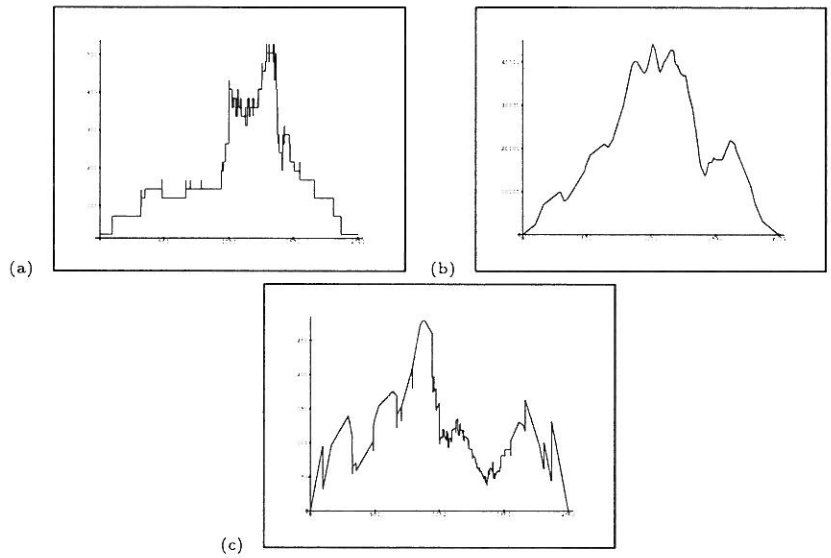
Le contraste relatif cumulé est donc:

$$RC(s) = \sum_{pour P_a, P_b adjacents} rc(C_a, C_b, s)$$

La fonction $AVCT(s)$ est définie comme le ratio du contraste relatif cumulé par le nombre de frontières détectés.

$$AVCT(s) = RC(s)/N(s)$$

Cette fonction représente le contraste moyen, observé aux abords des frontières détectées par un seuil s.



(a) nombre de frontières détectées en fonction du seuil, (b) contraste cumulé en fonction du seuil, (c) contraste moyen

FIG. 3.2 - Exemple de calcul à partir d'histogrammes.

La valeur s^* maximisant $AVCT(s)$ constitue donc le seuil de classification des courbures de l'ensemble étudié, voir l'exemple figure 3.2. Un avantage majeur de cette méthode, est que le seuillage est totalement piloté par les données et ne nécéssite à aucun moment de paramétrage humain.

Limites de la méthode

La méthode de Kohler ne permet pas de discriminer des objets ayant une courbure à fort gradient. Un autre problème peut se poser: les objets à gradient de courbure constant, pourront subir des divisions indésirables mais logiques. Enfin, les objets de révolution auront une fonction $AVCT(s)$ par paliers, voir les figures, ce qui pourra

induire des erreurs dans la détermination des seuils. C'est pourquoi, on determinera le seuil maximal sur un palier maximal comme la valeur moyenne sur ce palier.

3.2.4 Division d'objets par seuillage récursif sur les cartes

L'algorithme que nous proposons effectue un seuillage de Kohler *récursivement* sur les cartes.

Selection de l'adjacent de gradient maximal

La méthode décrite calcule et cumule les valeurs de $RC(s)$ et $N(s)$ pour toutes les faces adjacentes à une face courante. Or le but du seuillage est de trouver la valeur de courbure marquant le plus fort contraste. Le contraste sera à-priori plus important sur les frontières des cartes qu'à l'intérieur de celles-ci. Cependant, tant qu'une carte est sous-segmentée, il faut en extraire les frontières les plus robustes.
Afin de privilégier l'extraction des cartes intérieures les plus contrastées dans une carte donnée, on traite les faces intérieures, comme suit:
- pour mettre à jour les fonctions $RC(s)$ et $N(s)$, on sélectionne la face adjacente la plus différente en valeur de courbure. Cela permet en outre de réduire les possibles effets de superposition de données dûs à une taille trop importante de l'ensemble à étudier.

Contraintes de divisibilité pour les cartes

L'algorithme de seuillage est dirigé par les données. On impose cependant deux contraintes de divisibilité pour une carte:

- Sa taille doit être supérieure à une face.
- La valeur de $AVCS(s^*)$ décroit au fur et à mesure que l'on divise une carte. Son seuil $s*$ de division doit correspondre à des frontières suffisamment consistantes. Par conséquent, on impose $AVCS(s*) > 1$.

3.2.5 L'algorithme de seuillage automatique récursif sur les cartes

On note NC le nombre de courbures d'une face de l'objet. Il y en a deux pour une surface 3D : la courbure minimale et la courbure maximale. L'algorithme sera donc effectué NC fois sur l'objet. On aura donc pour une même carte traitée, deux seuils à

l'issue du calcul, un pour chaque courbure. Les nouvelles cartes sont déterminées ensuite en tenant compte de la connexité et de la valeur des courbures par rapport aux seuils calculés. Nous concerverons la notation NC plutôt que les indices 1 et 2 (on aurait pu aussi prendre min et max comme indices) parce qu'elle permet d'adapter la méthode a d'autres paramètres : pour les images par exemple, on a $NC = 3$ parce qu'il y a trois canaux couleur à prendre en compte.

A l'étape initiale $k = 1$ on dispose d'une seule carte qui est l'objet original, et pour laquelle on calcule les NC seuils de division.

Au début d'une étape $k > 1$, on dispose d'un ensemble d'étiquettes associées à chaque carte, appelées aussi labels.

Algorithme 3.2.5.1 **A l'étape k**, *pour chaque carte divisible A_{n_k} de l'objet, de seuils $s_{nc}(A_{n_k})$:*

1. *Calculer les NC seuils $s_{nc}(A_{n_k})$ comme suit :*

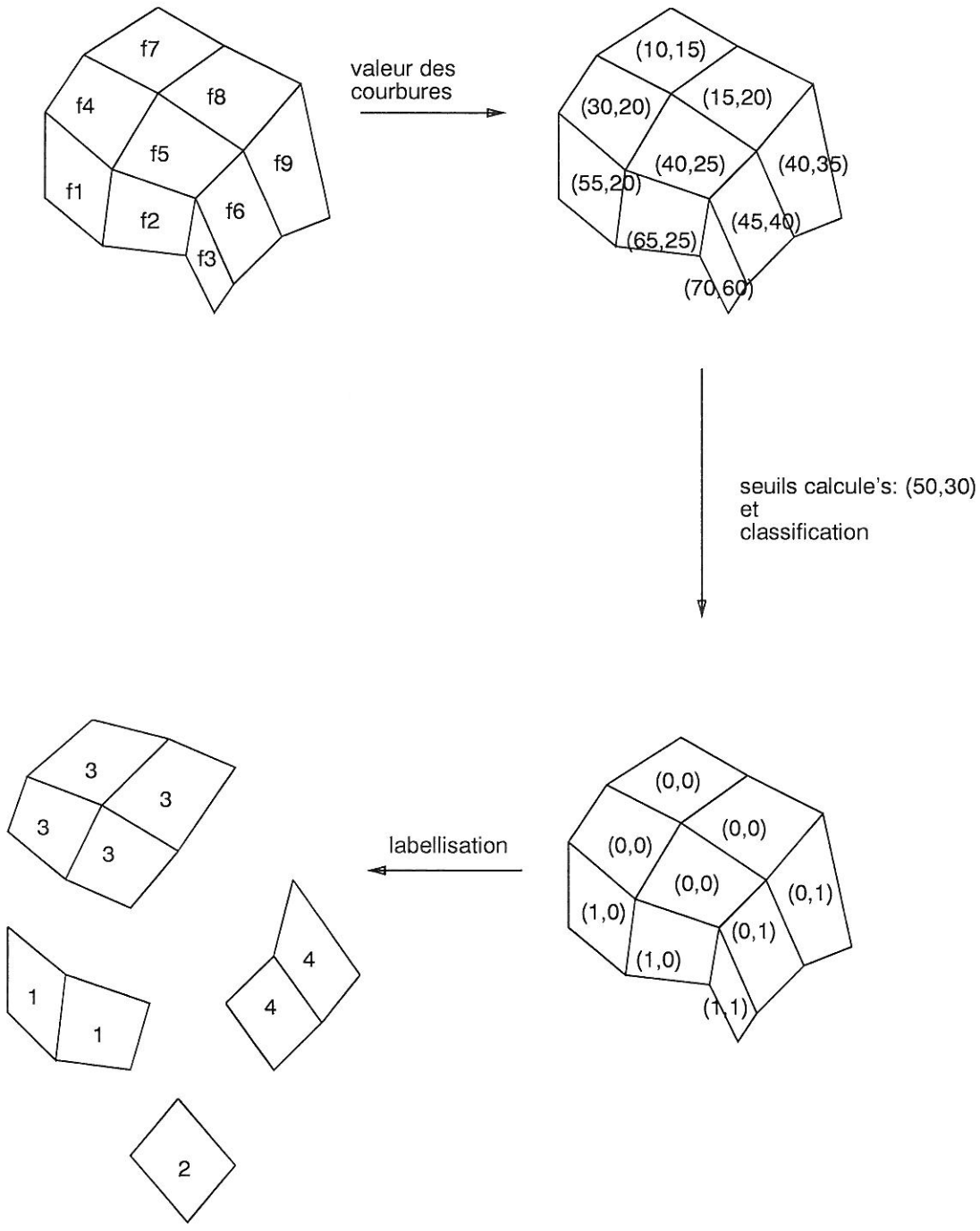
   - *calculer les NC fonctions $N(s)$ et $RC(s)$, en restreignant les calculs à la face adjacente la plus différente en courbure*
   - *Pour chaque composante suffisamment grande A_{n_k} extraire le(s) seuil(s) s^*_{NC} réalisant:*

     $$MAX[AVCS(s) = RC(s)/N(s)]$$

     - Si pour un des NC seuils on a $AVCS(s^*_{NC}) > 1$ *alors on retient les NC seuils.*
     - Sinon A_{n_k} *devient indivisible.*
   - *Les composantes de taille trop petite deviennent indivisibles*
2. *Affecter à chaque face f une classe parmi les 2^{NC} classes possibles suivant si*

   $$C_{nc}(f) < s_{nc}(A_{n_k}) \text{ ou } C_{nc}(f) >= s_{nc}(A_{n_k})$$

   remarque : dans le cas des courbures on aura quatre classes représentées par les couples (0,0),(0,1),(1,0),(1,1) où 0 signifie "en-dessous du seuil" et 1 "au-dessus du seuil".
3. *Construire les composantes connexes de chacune de ces classes*
4. *faire l'interface de ces données avec celles compatibles au modeleur : il faut remplir les structures de cartes et d'atlas pour pouvoir les traiter par des fonctions déjà concues par Jérome Maillot lors de sa thèse, notamment pour afficher ces cartes à l'écran (mais nous y reviendrons par la suite).*



Voici sommairement présenté, une segmentation de l'algorithme : soulignons que les valeurs ne sont pas réalistes et qu'elles ne font que servir d'illustration. Le premier croquis représente la surface considérée, elle est polygonalisée en facettes dont on donne le numéro.

FIG. 3.3 - Une étape de segmentation.

On peut suivre sur la figure 3.3 l'évolution du processus : on part d'une surface facettisée qui est en fait, à l'entrée de l'algorithme, une liste de faces et leur graphe d'adjacence, puis on récupère le résultat du calcul des courbures, on calcule deux seuils qui permettent de classifier les faces de l'objet, enfin à partir de cette classification, on regionne la surface en utilisant la connexité donnée par le graphe d'adjacence. On réitère ce processus sur les cartes obtenues jusqu'à ce qu'il n'y ait plus de cartes divisibles.

3.2.6 Utilisation et temps de calcul

En pratique

L'algorithme de division est itéré à la main c'est à dire que l'utilisateur doit commander l'étape d'itération. Ceci pour plusieurs raisons :

- on a des résultats exploitables dès les premières segmentations. La boucle récursive sur les cartes comporte peu d'itérations, on peut donc les suivre pas à pas.
- il est nécessaire de visualiser les cates obtenues après chaque segmentation.
- on peut ainsi arreter l'algorithme à tout moment lorsqu'on est satisfait de la segmentation (critère alors purement visuel). En effet, bien souvent et nous allons le voir dans les exemples, la première segmentation suffit.

Complexité et temps de calcul

Au niveau de la complexité, l'algorithme de segmentation descendant proposé se décompose en 4 parties, n étant le nombre total de faces de l'objet polygonalisé étudié :

- La récupération de la courbure (on ne parle pas du calcul de la courbure proprement dit) : on parcourt toutes les faces en $\Theta(n)$.
- Le calcul des histogrammes :on visite chaque face et on sélectionne la face la plus différente en valeur de courbure pour ensuite remplir l'histogramme. A priori, il s'agit d'un processus en $\Theta(np)$ avec p le nombre de faces adjacentes à une face, p dépend de n mais on peut borner sa valeur par 4 en pratique. Le coût est là encore de $\Theta(n)$. La détermination du seuil est constante, elle dépend uniquement de la précision voulue sur la courbure qui est déterminée en dehors de l'algorithme.
- La classification : affecter à chaque face la valeur 0 ou 1 selon le seuil se fait en $\Theta(n)$.

- Le calcul des régions : on visite chaque face et on vérifie si elle a déjà été traitée ; si c'est le cas, on ne s'en occupe plus ; sinon (son label est nul, elle n'a pas été traitée) on lui affecte le label courant incrémenté de 1, puis on cherche la première de ses faces adjacentes non déjà traitée qui possède la même classe de courbure (étape précédente), s'il en existe une, on réitère récursivement le processus avec le même label . Au pire des cas, à partir d'une seule face, on pourrait traiter tout le graphe d'adjacence, et il n'y aurait qu'une seule région. Cependant, ce processus reste en $\Theta(n)$ du fait de la vérification préalable du label (traité ou non traité) : dès que les n faces ont été traitées, le processus est terminé.

Finalement, la complexité de l'algorithme est fonction linéaire du nombre de faces.

L'algorithme précédent a été implémenté en langage C sur des Iris Silicon Graphics. Il a été intégré dans le modeleur ACTION 3D de l'INRIA. Pour un objet de 3000 faces, il calcule la segmentation et la mise à plat en une dizaine de seconde sur une IRIS 320VGX, mais c'est l'opération de mise à plat des cartes qui prend le plus de temps.

Chapitre 4

Résultats

Ce chapitre est consacré à la comparaison des deux algorithmes précédents à partir des segmentations qu'ils produisent, on analysera indépendamment la segmentation descendante et les améliorations qui ont pu y être apportées. Une dernière partie sera consacrée aux evolutions futures sur ce sujet et aux voies de recherche à explorer ainsi qu'aux applications envisageables.

4.1 Préambule

Avant de commencer l'analyse des résultats, il est nécessaire de préciser notre objectif. Il s'agit en effet de découper un objet polygonalisé en surfaces quasi-développables. Mais il ne faut pas perdre de vue qu'une découpe, notamment en ce qui concerne le plaquage de texture, reste très subjective ; même si l'automatisation du processus est poussée, l'utilisateur aura souvent besoin de faire quelques retouches pour lesquelles des fonctions adaptées à la manipulation d'atlas existent déjà, voir [Mai92]. Ici cependant, nous ne traiterons que la partie automatisation du processus.

4.2 Tests et analyses

4.2.1 Segmentation de formes simples

Dans un premier temps, pour s'assurer du bon fonctionnement de l'algorithme, nous l'avons testé sur des surfaces simples telles que des cônes ou des cylindres ; le cas de la sphère est un peu particulier parce qu'en théorie elle a une courbure constante en toutes ses faces. Ces objets sont tous fermés, ils ne comportent aucun trou (simplement

connexes au sens topologique).

Le cône et le cylindre

Ce que l'on voudrait pour le cône est une segmentation en deux cartes : la base d'une part et le chapeau développé d'autre part. Cette exigence est réalisée avec l'algorithme descendant dès la première étape de segmentation comme on peut le constater sur la figure.
De même le cylindre a été correctement découpé en trois cartes dès la première étape : les deux bases et la partie centrale bien développée. La texture a donc été plaquée sur ces cartes ; les coutures sont évidemment très visibles sur une texture de damier, voir 4.1.

En revanche, l'algorithme ascendant effectue une mauvaise segmentation sur ces objets, comme on peut le voir sur les figures : il sursegmente le cône en trois parties qui n'ont rien de très représentatif de la topologie de l'objet ; il effectue de même une découpe du cylindre en quatre cartes sans rapport avec la topologie réelle, voir 4.2.
Il ne faut pas oublier que pour cette méthode, c'est l'utilisateur qui stoppe la segmentation par le choix interactif du nombre de cartes.

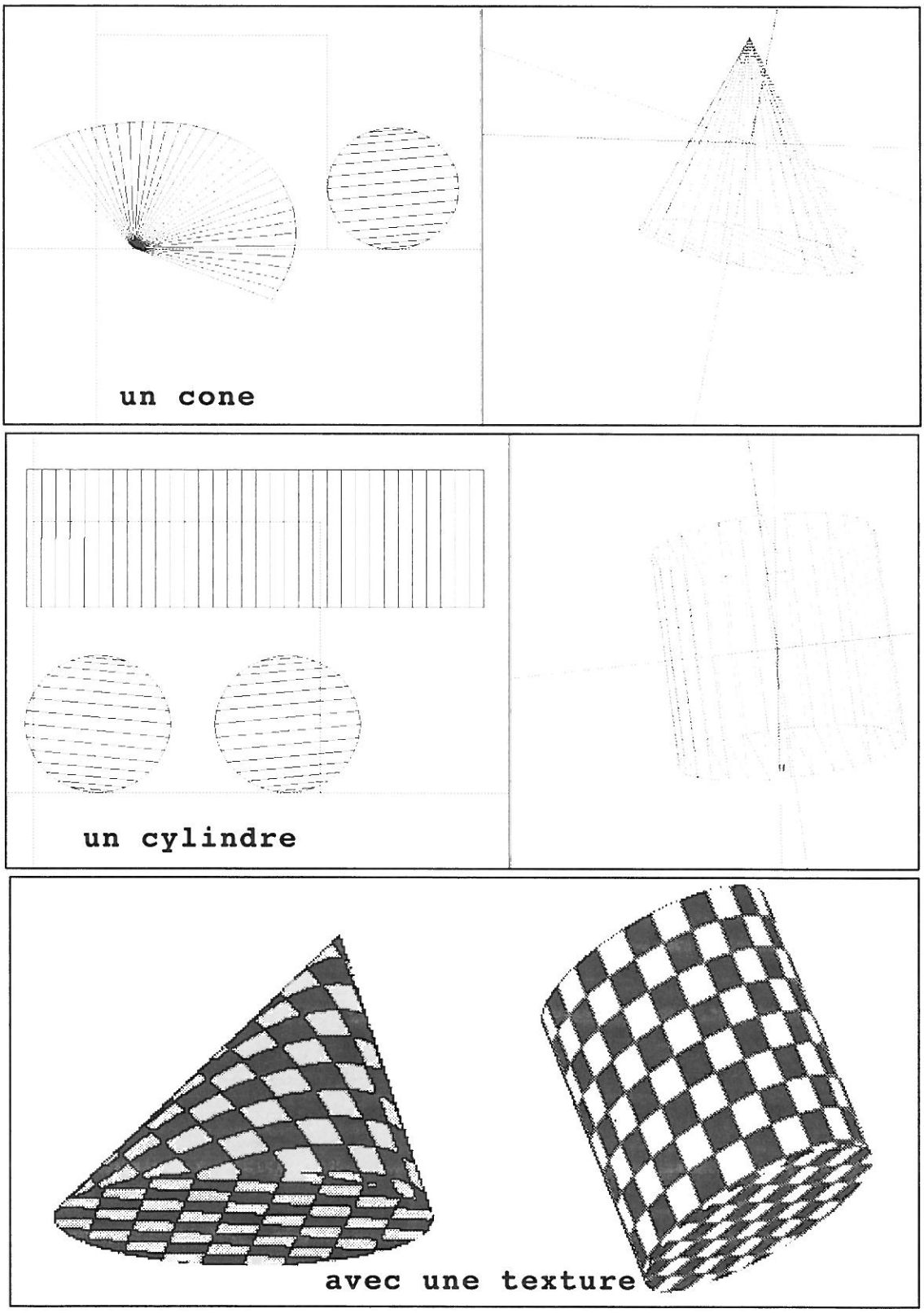
De fait, l'algorithme ascendant a été conçu pour faire une segmentation initiale pas forcément performante mais qui sert de base de travail à des fonctions interactives qui permettent d'améliorer cette segmentation (pour plus d'informations sur ces fonctions se référer à la thèse de Jérome Maillot [Mai92].
Nous ne poursuivrons donc pas la comparaison des deux algorithmes car l'optique dans laquelle ils ont été créés est finalement très différente : l'un est tourné vers l'interactivité, l'autre vers la création purement automatique. Nous ne nous interresserons désormais qu'à ce dernier qui est l'objet de cette étude.

Des formes un peu plus élaborées

Nous augmentons la difficulté en segmentant deux types de clous : le premier composé de formes simples, troncs de cônes et cylindres, et de 168 faces, l'autre composé de 6240 faces avec des formes plus continues.

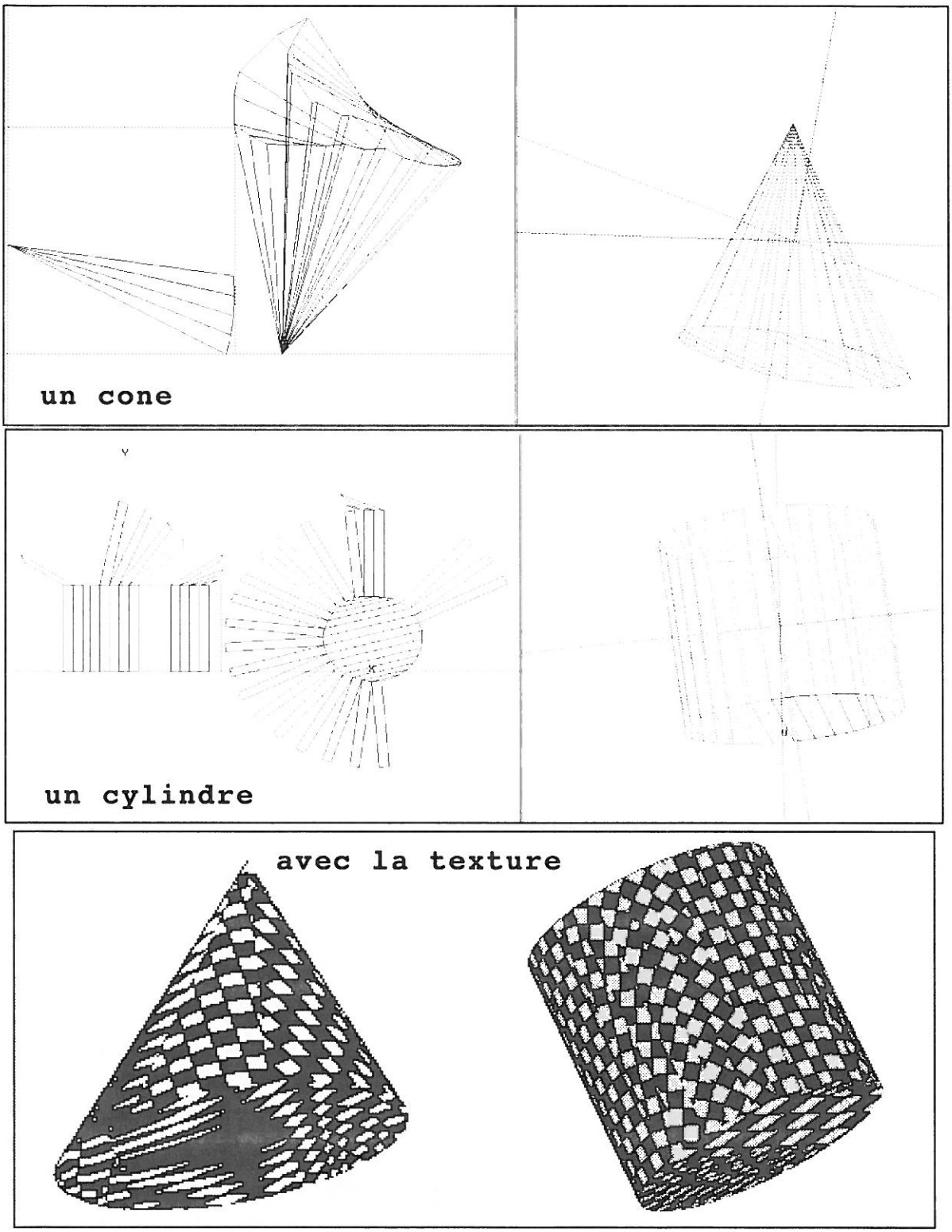
Sur la figure 4.3, la découpe s'effectue parfaitement et met en évidence les zones développables. Au bout de la deuxième étape, la segmentation stoppe d'elle même, les cartes étant devenues indivisibles.

Sur la figure 4.4, la segmentation laisse apparaitre des cartes très fines qui séparent en fait deux régions de courbures différentes. Nous avons là un effet du critère de labellisation des cartes pour lequel deux cartes different si un des seuils sépare une de leurs courbures respectives. Cet effet très discriminant du critère sera discuté et modifié



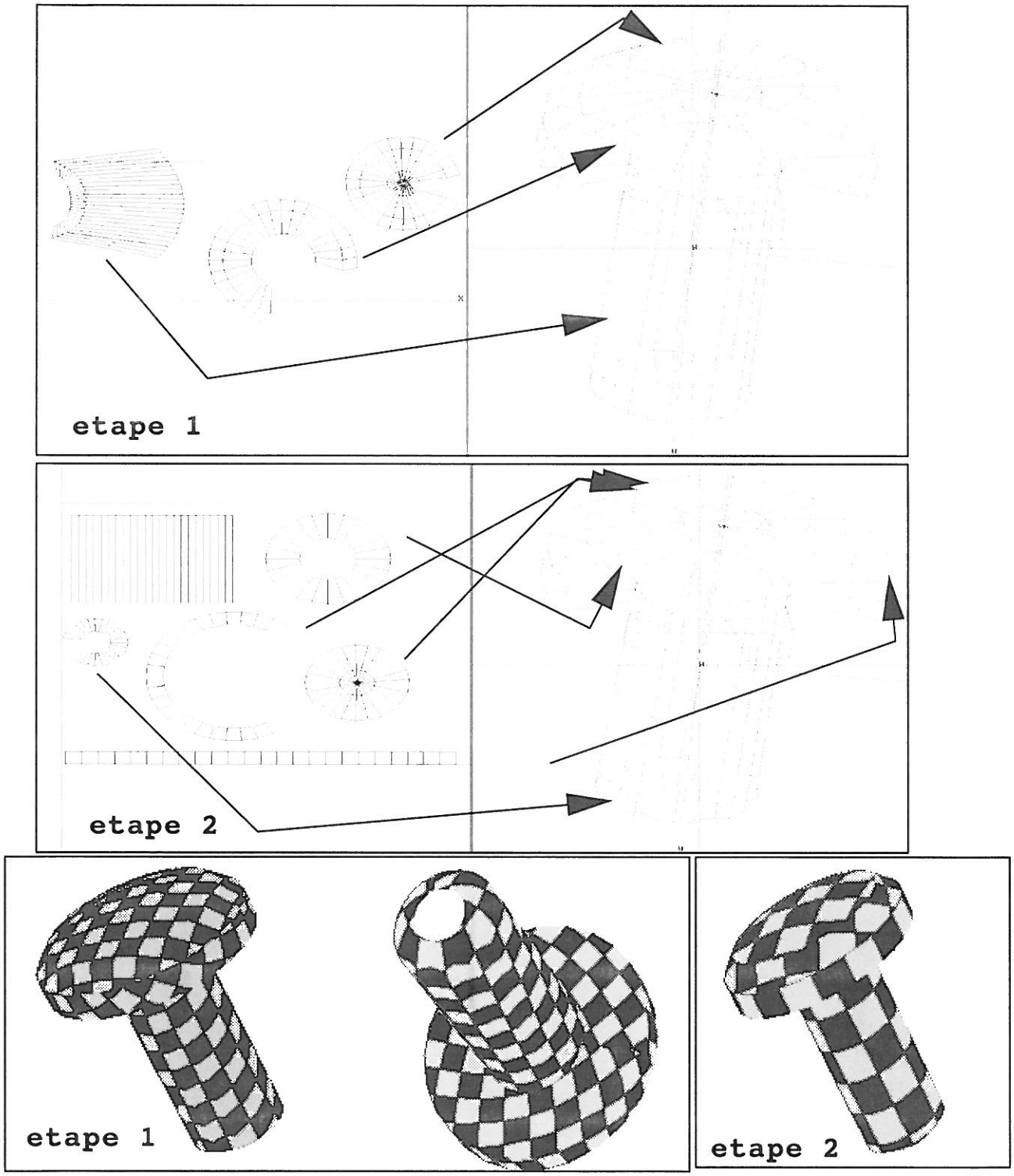
SEGMENTATION DESCENDANTE. Dans les deux premiers cadres, on a à gauche l'atlas des cartes mises à plat et à droite l'objet polygonalisé correspondant. Le cadre du bas représente les objets texturés. Ces images sont des saisies de l'écran du modeleur ACTION 3D.

FIG. 4.1 - Segmentation sur quelques objets simples.



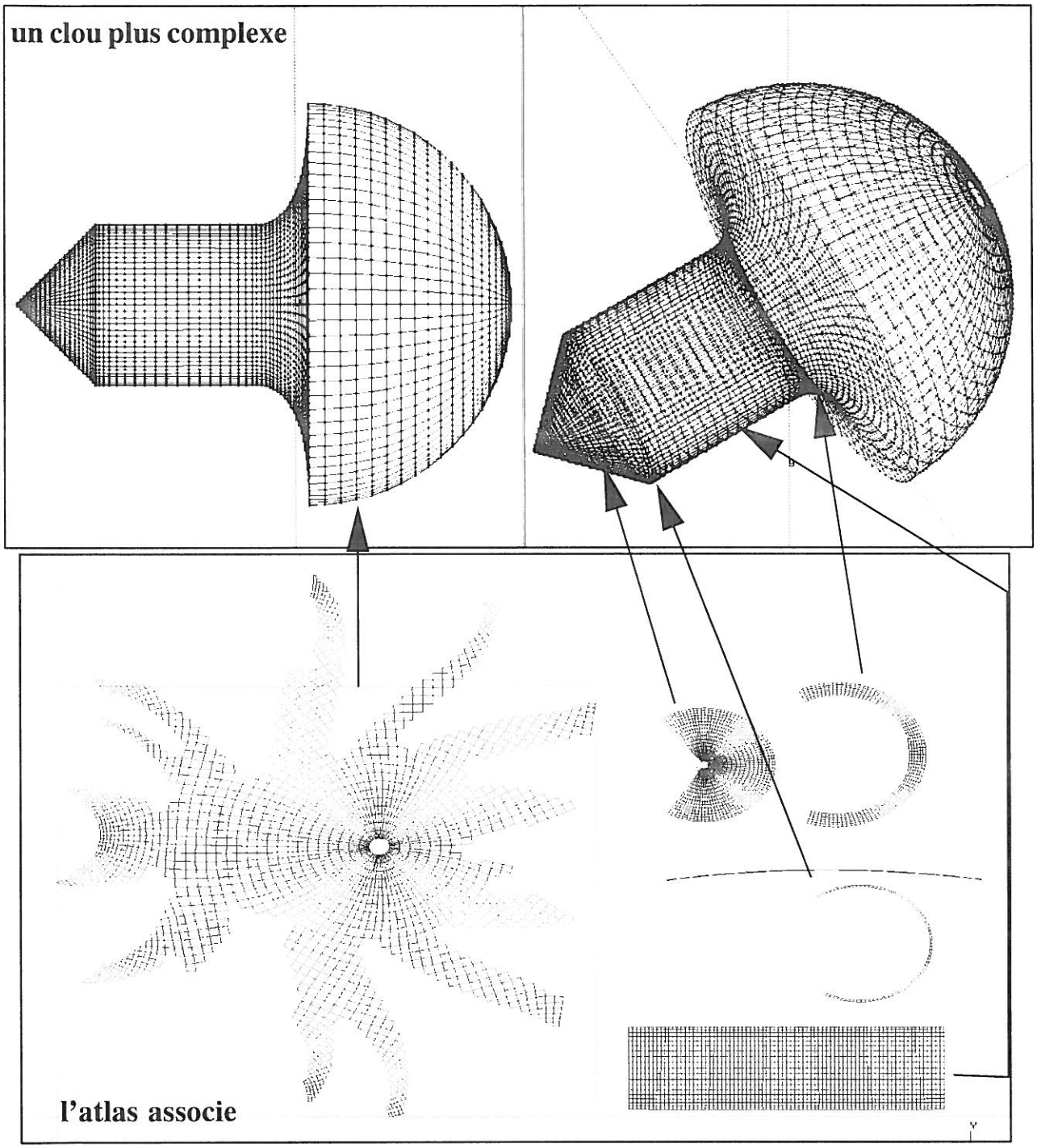
SEGMENTATION ASCENDANTE. Pour la comparaison, on a essayé de choisir le même nombre de cartes, la segmentation est beaucoup moins nette par rapport à la topologie de l'objet.

FIG. 4.2 - Segmentation sur quelques objets simples.



SEGMENTATION DESCENDANTE. Deux étape de segmentation découpent l'objet en toutes ses parties développables. Cependant, la première reste la plus intéressante pour le plaquage de texture. Les flèches indiquent quelles sont les parties de l'objet correspondant aux cartes.

FIG. 4.3 - Segmentation sur un clou simple.



SEGMENTATION ASCENDANTE. La segmentation reste bonne malgré l'émergence de deux cartes inconsistantes.

FIG. 4.4 - Segmentation sur un clou plus complexe.

par la suite. On peut considérer qu'il y a deux cartes de trop. Un autre problème survient : lors de la mise à plat des cartes, opération effectuée en dehors de la segmentation, un certain nombre de faces ne sont pas représentées à l'écran car elles dépasse le seuil de distorsion autorisé due à cette mise a plat. Cela explique la forme très éclatée de la carte la plus grande. Mais ceci n'empèche pas de savoir comment a été découpé l'objet.

Ces exemples sont assez bien découpés parce qu'ils sont composés de portions de surface parfaitement développables donc très adaptés à une segmentation basée sur la courbure, raison du bon fonctionnement de l'algorithme dans ces cas. Nous allons etudier maintenant un exemple beaucoup moins facile à traiter.

4.2.2 Le visage

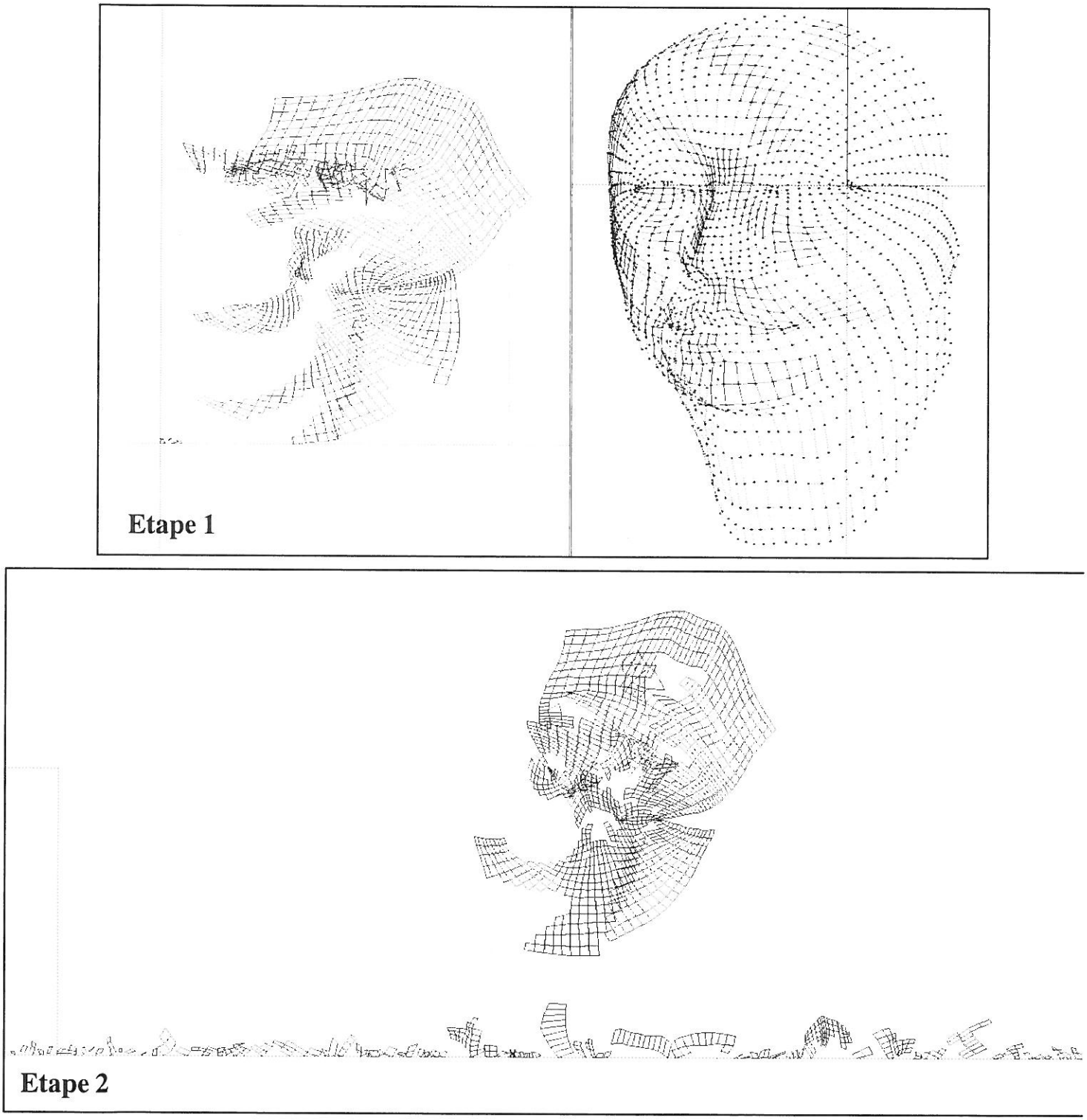
Ce modèle de 1536 faces d'un visage est très complexe à segmenter car les faces sont très diverses au niveau taille et courbures. Il s'agit d'un cas critique pour l'algorithme.

Essayons d'expliquer "physiquement" ce qui se passe. La méthode utilisée détecte le seuil de courbure sensé extraire le plus grand nombre de frontières les plus contrastées possible. Or un objet comme le visage présente des zones à faible gradient de courbure et des petites zones à fort gradient aux commissures des lèvres et des paupières par exemple. Ces petites zones très contrastées par rapport au reste "attirent" la valeur du seuil de courbure vers la leur, car il s'agit d'un contraste moyen, ce qui segmentera autour de ces petites zones, comme on peut le constater sur la figure 4.5.
Imaginons par exemple un objet difficile à concevoir qui possèderait un groupe de faces sensiblement de même valeur de courbure et une face de valeur de courbure très éloignée des autres, l'algoritme segmenterait dans un premier temps cette face isolée.

Dans le cas présent, le problème est le même. Regardons sur les histogrammes comment se produit cette aberration dans la détermination du seuil, voir figure 4.6, c'est très visible. A la deuxième étape, la segmentation est inutilisable dans cet état pour ce qui est du plaquage de texture : plaquer un damier sur une ou deux faces n'apporte rien. Plusieurs modifications peuvent être apportées pour essayer d'améliorer ces défauts de segmentation.

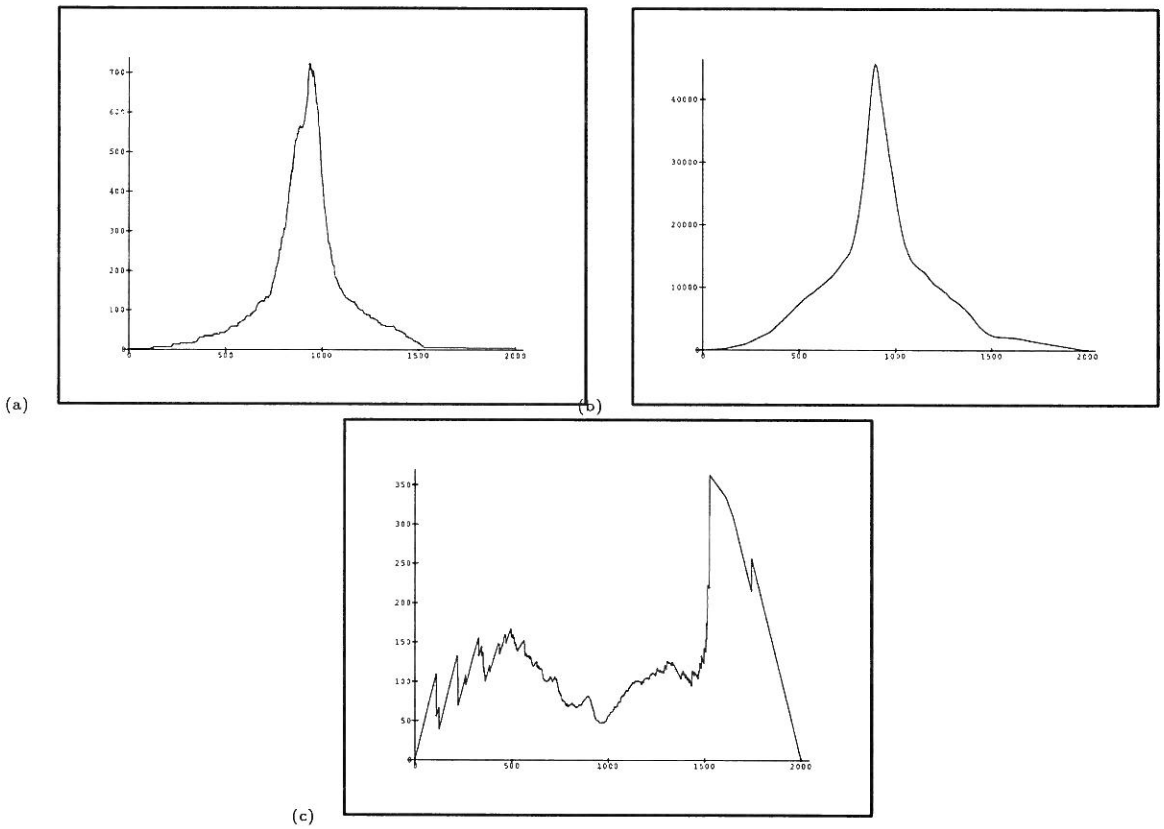
4.3 Quelques modifications

La question est toujours de savoir comment découper intuitivement un objet complexe tel qu'un visage en zones peu nombreuses et facilement développable ? Le problème reste ouvert. Ce que l'on peut en outre tenter, est de réduire le nombre de cartes de départ pour ensuite progressivement les redécouper.



SEGMENTATION ASCENDANTE. Le visage est une forme complexe qui met en défaut l'algorithme de segmentation.

FIG. 4.5 - Segmentation sur un visage.



(a) nombre de frontières détectées en fonction du seuil, (b) contraste cumulé en fonction du seuil, (c) contraste moyen. On remarque que le seuil est situé à droite du diagramme dans une zone de faible segmentation.

FIG. 4.6 - Histogrammes obtenus sur le visage.

Trois axes de modification viennent à l'esprit :

- Le type de paramètre traité : les courbures, les normales, la courbure gaussienne (qui est le produit des deux courbures précédemment employées), ...
- La création des régions : rendre le critère d'appartenance à une carte plus restrictif.
- Le calcul du seuil : imposer une taille minimum pour les cartes, et éliminer ainsi du calcul du contraste moyen les abérations de cacul que l'on a évoquées au chapitre précédent.

4.3.1 En entrée de l'algorithme

La modification que nous avons retenue ici concerne la courbure. En effet, la segmentation à partir des coordonnées de normales, donc sur trois composantes, donnerait plus de cartes et la sursegmentation n'en serait pas amoindrie.

Nous avons donc effectué une segmentation à partir de la courbure gaussienne qui est le produit des deux courbures utilisées jusqu'ici. Rappelons qu'une surface développable possède une courbure gaussienne nulle. Le seuil est donc calculé sur un seul paramètre. Le résultat est présenté sur la figure 4.7. On obtient effectivement moins de cartes dès la première étape, et l'on récupère les faces particulières des commissures des paupières qui contrastent très fortement sur le reste des faces. La deuxième étape sursegmente et l'on revient au problème précédent : cette découpe est inutilisable pour la texture.

4.3.2 En sortie de l'algorithme

Le critère d'origine pour la création des labels de cartes était le suivant : *deux faces appartiennent à une même carte si elles ont même classe à l'issue du seuillage et si elles sont adjacentes.* Par exemple toutes les faces de classe $(0, 0)$ et adjacentes forment une carte connexe.
Le critère modifié, plus restrictif, est le suivant : *deux faces adjacentes appartiennent à la même carte si elles ont toutes les deux la classe* $(0, 0)$ *ou toutes les deux la classe* $(1, 1)$ *ou toutes les une autre classe que ces deux dernières.*

On réduit donc le nombre de classes à trois. Cependant, comme précédemment, les faces des commissures des paupières sont segmentées dès le début et il y a trop de cartes ensuite, voir figure 4.8.

4.3.3 Taille des cartes

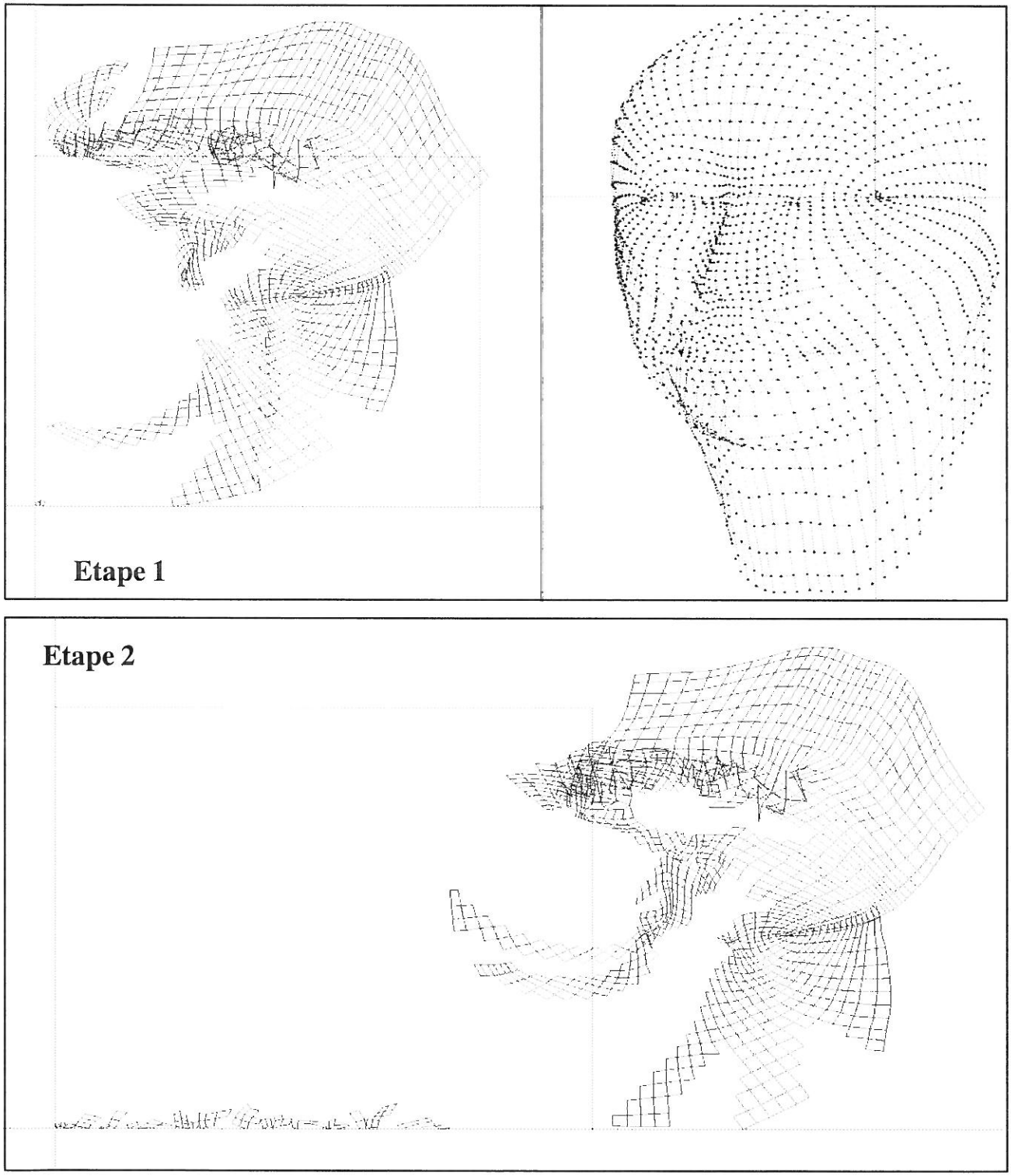


FIG. 4.7 - Segmentation à partir de la courbure gaussienne.

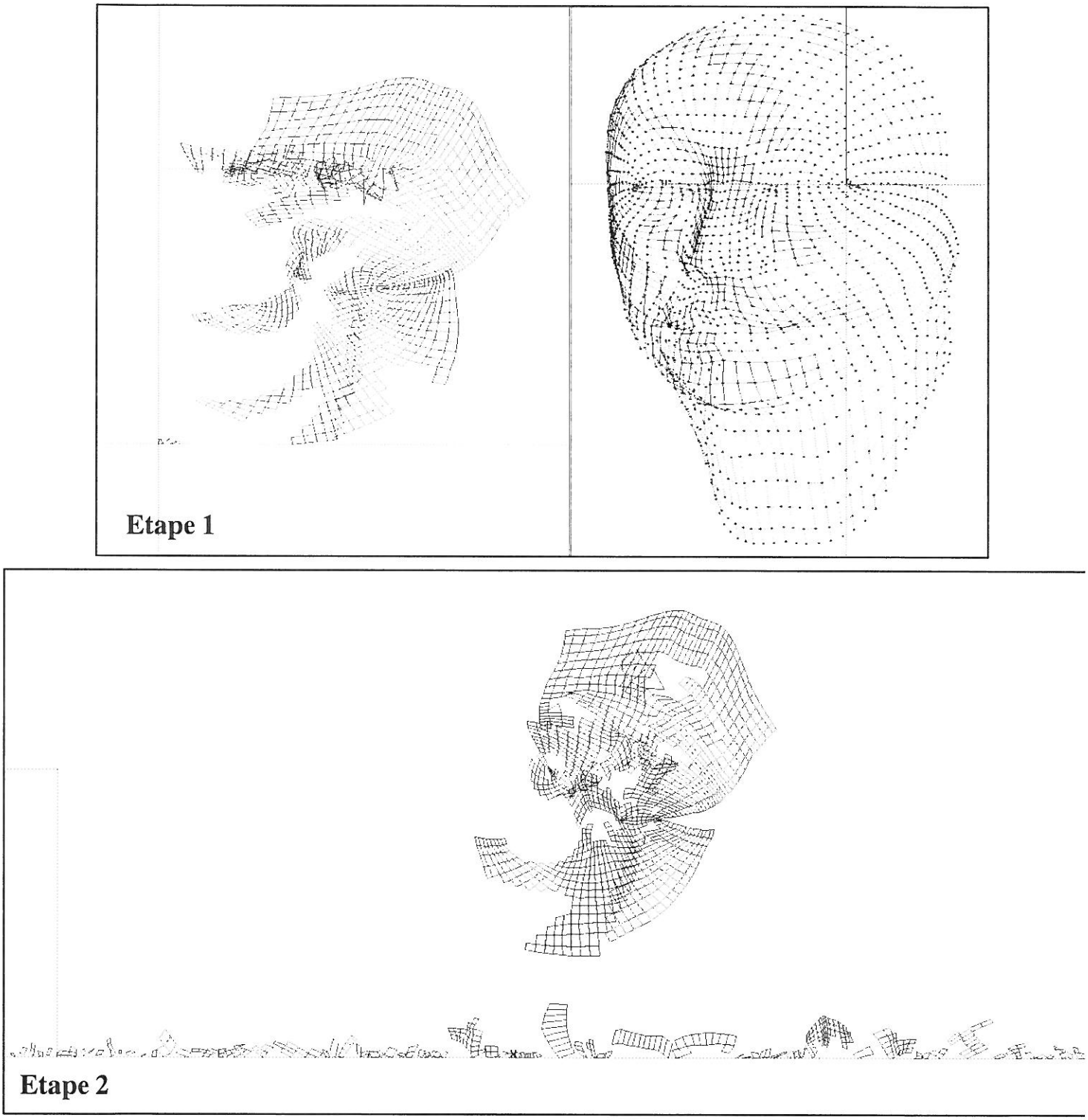


FIG. 4.8 - Segmentation sur un critère plus strict.

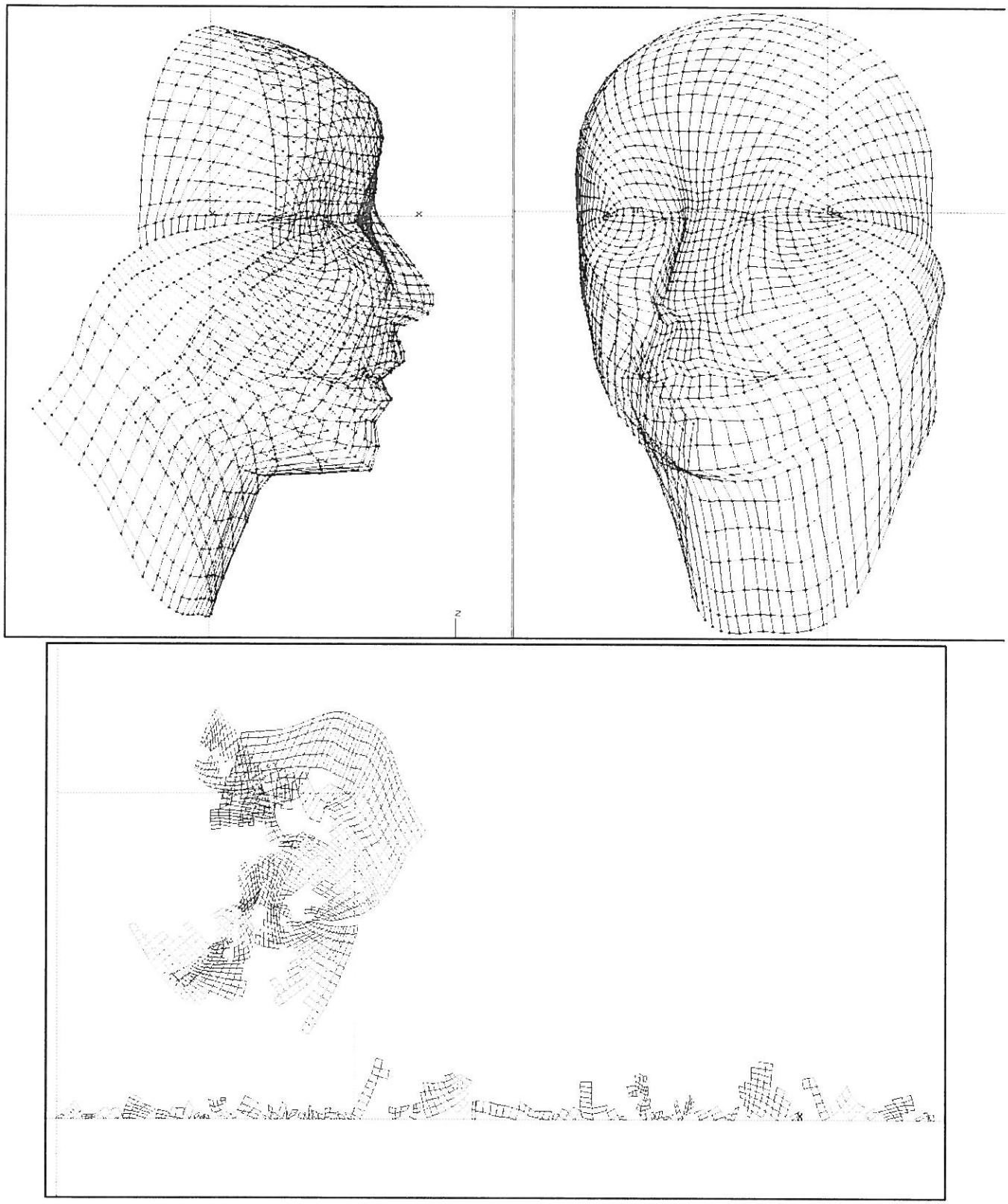


FIG. 4.9 - Réduction de la taille des cartes.

On veut maintenant jouer directement sur la détermination du seuil en évitant une segmentation dès la première étape trop faible (voir les exemples précédents). On impose donc la condition suivante : *le calcul du contrate moyen pour un seuil donné est effectué seulement si la valeur du nombre de frontières détectées est supérieure à une certaine taille définie par l'utilisateur.* Cette taille ne doit pas être déterminée n'importe comment : elle doit forcément dépendre du nombre total de faces de l'objet ; nous avons choisi de la prendre égale à la racine carrée du nombre total des faces en pensant à la découpe d'un morceau de toile carrée qu'on divise en deux, la frontière de séparation a la longueur d'un des côtés.

La segmentation dès la première étape apparaît sur la figure 4.9 : on a effectivement une segmentation un peu moins anarchique que précédemment, mais là encore rien de vraiement concluant pour ce qui est de la texture.

4.3.4 Conclusion

L'algorithme de seuillage récursif n'est donc pas adapté à la découpe en cartes d'objets complexes comme le visage, avec comme paramètre d'entrée la courbure. Par contre, il segmente bien les objets composés de portions de surfaces développables. Les vêtements sont bien souvent fait de bouts de cylindres cousus entre eux. Sur de tels objets, l'algorithme est alors performant.

Cependant un problème très important subsiste : celui de la mise à plat des cartes. Elle est certes optimale pour des surfaces développables, mais ne convient plus lorsque la segmentation est défaillante. La segmentation doit en quelques sortes préparer le travail de la mise à plat en découpant le mieux possible, d'où l'intérêt d'une bonne segmentation. Pour pallier cette difficulté est prévue une fonction d'optimisation de l'énergie de déformation ; cependant, cette technique n'améliore pas la segmentation, elle évite les déformations trop importantes au moment du plaquage de texture. Voyons maintenant quelles peuvent être les solutions à envisager.

4.4 Perspectives

Nous voyons deux voies à explorer : l'une issue des travaux de Y. Kergosien sur la topologie différentielle et la théorie des catastrophes ; l'autre, en considérant les prétraitements possibles sur l'objet en lui affectant des paramètres liés à l'énergie de déformation.

4.4.1 Solution topologique

Les travaux de Yannick Kergosien sur la topologie différentielle [Ker81] et [Ker92] nous permettent d'entrevoir, à partir de la recherche et de la localisation des singularités sur les surfaces 3D, une mise à plat plus efficace des cartes préalablement segmentées en les découpant partiellement. On remarque en effet que ces singularités sont la cause de déformations indésirables.

Le problème réside dans le fait que ces travaux n'ont, à notre connaissance, pas fait l'objet d'une étude dans le domaine du discret, une surface polygonalisée par exemple, et qu'il faut adapter les résultats mathématiques formels à ce domaine.

4.4.2 Utilisation de l'énergie de déformation

Puisque le critère de déformation est une énergie qu'on minimise, pourquoi ne pas l'utiliser dès la segmentation comme critère de sélection des cartes, plutôt que séparer les deux opérations ?

Nous voyons ainsi la possibilité d'affecter à chaque facette de l'objet, non pas sa courbure comme précédemment, mais la valeur calculée de l'énergie de déformation sur un voisinage centrée sur la facette considérée. Puis, par un algorithme semblable à celui décrit dans ce compte rendu, nous pourrions segmenter directement l'objet en régions de même énergie de déformation, et de fait précalculer la mise à plat.

Chapitre 5

Conclusion

La technique de segmentation descendante par seuillage récursif exposée précédemment n'avait jamais été utilisée auparavant pour travailler sur des objets de synthèse. Elle a permis une amélioration de la segmentation automatique par rapport à ce qui existait, et reste une aide efficace aux fonctions de traitement des atlas déjà mentionnées. Elle n'est pas idéale car aujourd'hui encore il est difficile, non seulement de concevoir un algorithme de création d'atlas entièrement automatique, mais aussi de pouvoir dire objectivement ce que l'on voudrait dans tel ou tel cas de figure. Néanmoins, elle permet de détecter les surfaces développables, s'il y en a, au risque même de sursegmenter d'autres cartes.

Quoiqu'il en soit, bien que des modifications soient nécessaires, cet algorithme est un outil très utile du fait de ses propriétés intrinsèques d'abord et qu'il puisse traiter n'importe quel paramètre sur les faces, ce qui rendent les utilisations possibles très nombreuses.

Annexe A

Calcul de la courbure.

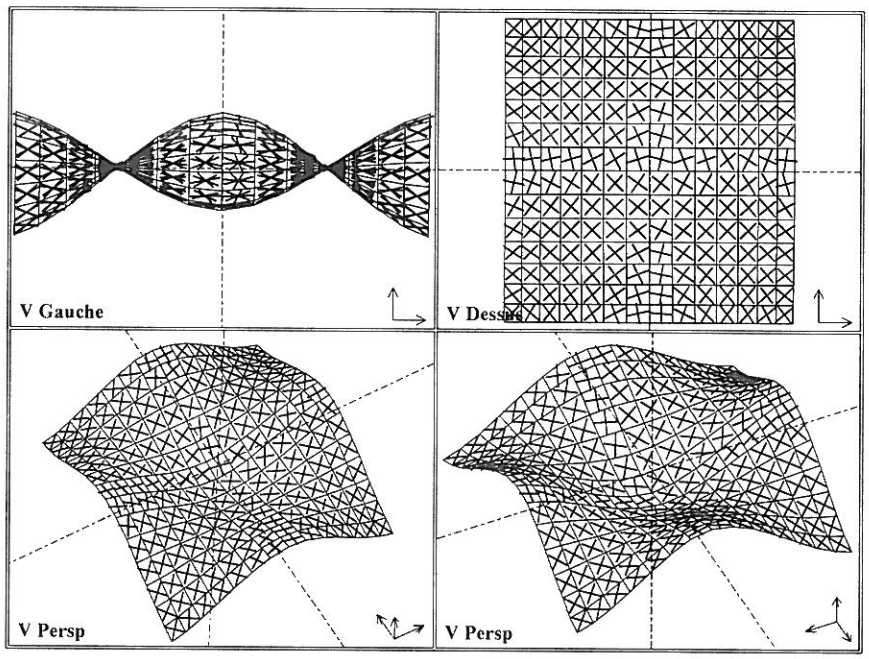
La courbure en un point est une forme quadratique du plan tangent. On peut la définir à partir de la différentielle de l'application de Gauss $\underline{\mathsf{N}}$, en identifiant par une translation le plan tangent à la surface et celui tangent à la sphère unité au point correspondant[DoC76, pages 135-151], par la formule suivante, avec u un vecteur du plan tangent:

$$C(u) = u \cdot \mathrm{d}\underline{\mathsf{N}}(u) \tag{A.1}$$

Il existe des algorithmes précis mais coûteux pour calculer la courbure d'une surface discrétisée [SaZu87]. Puisque ce calcul est destiné à servir de point de départ à un algorithme plus complexe, et que le temps de réponse est une de nos principales préoccupations, nous avons commencé par écrire un algorithme naïf qui s'avère suffisamment précis.

Nous utilisons une méthode de différences finies pour approximer les dérivées. Pour cela nous devons mesurer la normale en au moins trois points proches et non alignés afin de définir les trois coefficients de la matrice de courbure. Le plus simple est d'utiliser les normales lissées aux sommets que l'on calcule pour le rendu, qui sont définies en chaque sommet comme la moyenne des normales des faces adjacentes [Bre88, page 245]. Il serait aussi possible de prendre directement les normales des faces voisines, mais une phase de lissage permet de diminuer le bruit, en particulier pour les objets issus d'un scanner.

Nous affectons la normale N de la face en son centre de gravité G. Nous possédons alors un point G et une normale N de référence, ainsi qu'une collection de sommets S_i associés aux normales N_i. A l'aide de ces couples nous allons chercher à évaluer la différentielle $\mathrm{d}\underline{\mathsf{N}}$. On doit alors trouver une application linéaire symétrique $\mathcal{L}$ sur le plan tangent $\mathcal{P} = (G, \{N\}^{\perp})$ telle que pour tout i, $\mathcal{L}(\overrightarrow{GS_i})$ se rapproche le plus de $N_i - N$. La formule A.1 montre que la matrice de l'application linéaire $\mathcal{L}$ est aussi la matrice de courbure.



On voit ici un exemple de visualisation interactive des lignes de courbure. Ces lignes ne sont pas calcule'es explicitement mais la visualisation des directions propres au centre des faces suffit pour arriver a' les imaginer. A l'e'cran, on utilise deux couleurs diffe'rentes pour distinguer les courbures positives des courbures ne'gatives. Il est aussi possible de n'afficher que les directions de courbure maximale ou minimale, ou de donner des longueurs proportionnelles a' la courbure. On remarquera les discontinuite's aux voisinage des ombilics (en particulier au centre de la surface).

FIG. A.1 - Visualisation des lignes de courbures.

Pour s'affranchir du cas où la face considérée n'est pas plane, on commence par projeter toutes les différences $N_i - N$ dans le plan $\mathcal{P}$. On recherche ensuite les coefficients de la courbure par une méthode des moindres carrés. On appelle n_i et s_i les normales et sommets projetés. On remarque que par définition de $\mathcal{P}$, la projection de N est nulle. On écrit la matrice de courbure dans une base du plan tangent sous la forme $C = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$. Les points n_i et s_i sont décrits dans la même base, avec G comme origine. On doit donc minimiser :

$$\sum_i ||Cs_i - n_i||^2 = \sum_i \left\| \begin{pmatrix} s_i^x & s_i^y & 0 \\ 0 & s_i^x & s_i^y \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} - \begin{pmatrix} n_i^x \\ n_i^y \end{pmatrix} \right\|^2 \tag{A.2}$$

On pose $\sigma_x = \sum s_i^{x2}$, $\sigma_y = \sum s_i^{y2}$ et $\sigma_{xy} = \sum s_i^x s_i^y$, la solution s'écrit alors :

$$d = (\sigma_x + \sigma_y)\left(-{\sigma_{xy}}^2 + \sigma_x\sigma_y\right) \tag{A.3}$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \frac{1}{d} \begin{pmatrix} -{\sigma_{xy}}^2 + \sigma_x\sigma_y + {\sigma_y}^2 & -\sigma_{xy}\sigma_y & {\sigma_{xy}}^2 \\ -\sigma_{xy}\sigma_y & \sigma_x\sigma_y & -\sigma_x\sigma_{xy} \\ {\sigma_{xy}}^2 & -\sigma_x\sigma_{xy} & {\sigma_x}^2 - {\sigma_{xy}}^2 + \sigma_x\sigma_y \end{pmatrix} \cdot \begin{pmatrix} \sum s_i^x n_i^x \\ \sum s_i^x n_i^y + s_i^y n_i^x \\ \sum s_i^y n_i^y \end{pmatrix}$$

Les vecteurs propres v et les valeurs propres α sont alors donnés par :

$$v_\pm = \begin{pmatrix} 2b \\ c - a + \Delta \end{pmatrix} \quad , \quad \alpha_\pm = \frac{1}{2}(\mathrm{tr}(C) + \Delta) \tag{A.4}$$

(A.5)

avec:

$$\Delta = \pm\sqrt{\mathrm{tr}(C)^2 - 4\det(C)}$$

Les valeurs propres sont respectivement appelées courbure minimale et courbure maximale, celles qui sont utilisées en entrée de l'algorithme. Cette méthode de calcul est bien plus rapide qu'un algorithme fin basé sur une approximation de la surface par une quadrique [SaZu87]. Il a donné des résultats suffisamment précis pour tous les tests que nous avons pu faire, que ce soit pour des objets de synthèse, ou des objets digitalisés.

Annexe B

Une énergie utilisant les distances et les surfaces orientées

L'énergie totale se calcule sur l'objet polygonalisé avec des triangle. Elle est de la forme :

$$E_{\text{tot}} = (1-\alpha)\sum_{i}\sum_{j\in \text{Vois}(i)} \frac{\left(\left\|\overrightarrow{m_i m_j}\right\|^2 - \left\|\overrightarrow{M_i M_j}\right\|^2\right)^2}{\left\|\overrightarrow{M_i M_j}\right\|^2} + \alpha \sum_{\text{triangles(i,j,k)}} \frac{\left(\det(\overrightarrow{m_i m_j}, \overrightarrow{m_i m_k}) - \left\|\overrightarrow{M_i M_j} \wedge \overrightarrow{M_i M_k}\right\|\right)^2}{\left\|\overrightarrow{M_i M_j} \wedge \overrightarrow{M_i M_k}\right\|} \quad \text{(B.1)}$$

Le premier terme est l'énergie calculée sur les distances, le second l'énergie sur la surface. Le déterminant est calculé dans la plan de la texture. On peut remarquer que le terme $a = \det(\overrightarrow{m_i m_j}, \overrightarrow{m_i m_k})$ est bilinéaire par rapport aux coordonnées des points, alors que le terme $A = \left\|\overrightarrow{M_i M_j} \wedge \overrightarrow{M_i M_k}\right\|$ fait intervenir une racine carrée que l'on ne peut pas supprimer. Le gradient de E_2 est dont beaucoup plus simple par rapport au points m_i dans l'espace texturel que par rapport au points M_i dans $\mathbb{R}^3$.

On a supposé implicitement que la surface est orientable et que les triangles $M_i M_j M_k$ sont décrits dans le sens direct par rapport à la normale de la face. En effet, A représente une aire positive, alors que a est une aire orientée. C'est donc le signe de a qui permet de contrôler l'orientation de la mise à plat des triangles.

On peut aussi remarquer la normalisation de l'énergie en $\left\|\overrightarrow{M_i M_j} \wedge \overrightarrow{M_i M_k}\right\|^{-1}$. En

effet, si l'on subdivise une face en quatre parties égales par une homothétie de rapport $\frac{1}{2}$, la somme des quatre termes est identique à l'énergie de la face initiale.

Le gradient de cette fonction est toujours un polynôme de degré trois par rapport aux coordonnées des points m_i. On peut optimiser E_{tot} par une méthode de gradient conjugué.

Le coefficient α est déterminé au niveau des paramètres globaux du modeleur.

Annexe C

Présentation de l'Institut

Fondé en 1967, L'INRIA (Institut Nationnal de Recherche en Informatique et en Automatique) est un EPST (Etablissement Public à caratère Scientifique et Technologique), consacré à l'informatique et à l'automatique. L'INRIA est placé sous la tutelle des ministères de la Recherche et de l'Industrie, et est régi par les règles de la Fonction Publique. L'Institut est localisé sur cinq sites qui sont Rocquencourt, Lorraine, Rennes, Grenoble et Sophia Antipolis.

L'INRIA compte près de 1300 personnes, dont 1000 scientifiques répartis en 250 chercheurs permanents, 50 ingénieurs de l'industrie, 180 chercheurs d'autres laboratoires du secteur public, 100 chercheurs étrangers invités, 330 stagiaires et boursiers (dont 110 venant de l'étranger).

L'Institut est doté d'un budget de 445MF TTC, dont 20% de ressources propres provenant essentiellement de contrats de commercialisation de logiciels (LE-LISP, MENTOR, SABRINA, EDIMATH, SICLA, BASILE, etc.), de contrats de recherche soit avec des organismes publics (DRET, CNET, MRES, CNES, IFREMER), soit avec des sociétés privées (Aérospatiale, Thomson, Renault, etc.) et de contrats de recherche avec la CEE (ESPRIT, MAP).

Les missions de l'INRIA sont la recherche fondamentale et appliquée, la réalisation de systèmes expérimentaux, la valorisation des résultats, la diffusion des connaissances, les échanges scientifiques internationaux, la contribution à des programmes de coopération, les expertises scientifiques, et la contribution à des actions de normalisation.

Dans un rôle de prospective et d'expertise, l'INRIA intervient dans la réalisation de développements industriels avancés d'intérêt général, en collaboration avec des industriels, dans l'évaluation de matériels mis à disposition par les constructeurs (Pyramid, NP1, ETA 10-P, MIPS,..), dans l'expérimentation d'architectures nouvelles, dans l'expertise auprès d'organismes publics ou industriels, dans la mise en oeuvre d'un plan de communication international et dans la gestion des passerelles internationales

(FNET, NSFNET, Y-NET, ARISTOTE).

L'Institut possède de gros moyens informatiques, son parc informatique est composé de

- 300 micro-ordinateurs,
- 900 stations de travail sous UNIX,
- 16 machines de service (Bull,Gould..),
- 3 Machines de "haute performance",
- 2 Convex C2, accès Cray II,
- des machines d'expérimentation: SEQUENT, IPSC, T-NODE, CONNECTION-MACHINE, KSR.

L'Institut est ouvert sur le monde entier grâce à un réseau composé d'accès X25 (12 commutateurs), ETHERNET (1000 raccordements), HYPERCHANNEL (2 connexions), FDDI (14 accès ETHERNET). De plus l'INRIA gère quatre passerelles: FNET, ARISTOTE, NSFNET, Y-NET.

La création d'entreprises de haute technologie constitue un secteur de transfert privilégié de l'innovation. Depuis 1984, les chercheurs de l'INRIA ont ainsi créé 19 sociétés de ce type: 4 filiales de l'institut (Ilog, Simulog, 02 Technologies, Connexité) et 15 start-up (Aleph-Technologies, Chorus-Systèmes, Cose, Infosys, Ergomatic Consultants, Istar, Lorin, Euroclid, Noésis, Finaki, Robosoft, Gipsi S.A., Timeat, Grif, Verilog).

L'INRIA compte une soixantaine d'équipes de recherche réparties en six programmes.

1. Architectures parallèles, bases de données, réseaux et systèmes distribués
2. Calcul symbolique, programmation et génie logiciel
3. Intelligence artificielle, systèmes cognitifs et interaction homme-machine
4. Robotique, image et vision
5. Traitement du signal, Automatique et productique
6. Calcul scientifique, modélisation et logiciel numérique

Les diverses équipes sont réparties sur les différents sites. L'INRIA Rocquencourt est la seule des cinq unités où tous les programmes de recherche sont représentés. Le site de

Rocquencourt, le plus ancien (1967, s'appelait IRIA , puis INRIA en 1980), acceuille le siège de l'INRIA. 26 équipes de recherche travaillent dans ce seul site, soit environ 500 personnes.

Le projet SYNTIM (SYNthèse d'IMage), dirigé par André Gagalowicz occupe une vingtaine de personnes travaillant en trois équipes respectivement sur l'analyse, la synthèse d'images et la cartographie. Ce projet s'inscrit dans le programme 4 : robotique, image et vision. Les deux premières équipes, analyse/synthèse d'images travaillent sur une boucle fermée Analyse/Synthèse, afin de créer des images de synthèse aussi proches de la réalité que possible, et réciproquement analyser une scène le plus précisément possible.